



TITLE:

# Learning Weak Monadic Second-Order Logical Formulas from Queries and Counterexamples(Complexity Theory and Related Topics)

AUTHOR(S):

Nishino, Tetsuro

---

CITATION:

Nishino, Tetsuro. Learning Weak Monadic Second-Order Logical Formulas from Queries and Counterexamples(Complexity Theory and Related Topics). 数理解析研究所講究録 1990, 716: 37-59

ISSUE DATE:

1990-03

URL:

<http://hdl.handle.net/2433/101758>

RIGHT:

## 弱単項二階論理式の例示および反例からの学習

Learning Weak Monadic Second-Order Logical Formulas from Queries and Counterexamples

西野哲朗

Tetsuro Nishino

東京電機大学 理工学部 情報科学科

Department of Information Sciences, Tokyo Denki University

### Abstract

In this paper, we consider the problem of learning an unknown weak monadic second-order ( WMS for short ) logical formula from examples making it true and those making it false. It is well known that, for a WMS formula, there exists a deterministic frontier-to-root tree automaton ( dfrta for short ) accepting the trees which are the encodings of those examples making the formula true. Thus, we reduce our problem to the problem of learning an unknown tree language from examples of its members and nonmembers. We assume that the tree language is presented by the Angluin's *minimally adequate teacher*, which can answer membership queries and equivalence queries. Our learning algorithm  $L_T^*$  to learn tree languages is based on a slight variant of the Sakakibara's polynomial time algorithm to learn deterministic skeletal automata. The algorithm  $L_T^*$  runs in time polynomial in the number of states of the minimum dfrta for the unknown tree language and the maximum size of any counterexample provided by the teacher. Thus we propose a new framework of concept learning in which atomic relations such as " $\subseteq$ " and many relations definable by WMS formulas, e.g. prefix-closedness of sets, can be learned in polynomial time.

## 1 Introduction

In the theoretical studies of concept learning, the problems of learning various types of logical formulas have gained a great deal of attention in recent years. Many works have been performed in this area. For example, Shapiro showed an algorithm solving the model inference problem for Horn theories [11]. In [14], Valiant provided a theoretical basis for learning boolean formulas. While Angluin showed a polynomial time algorithm to learn acyclic propositional Horn sentences using equivalence queries and requests for hints. [2].

In this paper, we consider the problem of learning an unknown weak monadic second-order ( WMS for short ) logical formula from examples making it true and those making it false. And we will show a polynomial time algorithm to learn WMS formulas using membership queries and equivalence queries. Thus we propose a new framework of concept learning in which atomic relations such as " $\subseteq$ " and many relations definable by WMS formulas, e.g. prefix-closedness of sets, can be learned in polynomial time.

It is known that, for a WMS formula, there exists a deterministic frontier-to-root tree automaton ( dfrta for short ) accepting the trees which are the encodings of those examples making the formula true [13]. Thus, we reduce our problem to the problem of learning an unknown tree language from examples of its members and nonmembers. We assume that the tree language is presented by the Angluin's *minimally adequate teacher*, which is explained in the next paragraph.

Concerning the problem of learning an unknown regular set from examples, Angluin introduced a learning protocol so called *minimally adequate teacher* which can answer membership queries and equivalence queries [1]. Angluin showed that if an unknown regular set is presented by a minimally adequate teacher, the regular set can be learned in time polynomial in the number of states of the minimum deterministic finite automaton for the unknown regular set and the maximum length of any counterexample provided by the teacher.

In [9, 10], in order to develop an efficient learning algorithm for context-free grammars, Sakakibara reduced the problem of learning a context-free grammar from structural data to the problem of learning a deterministic skeletal automaton, which is a particular kind of dfrta. Then extending the Angluin's learning algorithm for finite automata to the one for deterministic skeletal automata, he obtained an efficient learning algorithm for context-free

grammars. Our learning algorithm  $L_T^*$  to learn dfrtas is based on a slight variant of this Sakakibara's polynomial time algorithm. The algorithm  $L_T^*$  runs in time polynomial in the number of states of the minimum dfrta for the unknown tree language and the maximum size of any counterexample provided by the teacher.

The paper is organized as follows: In section 2 we review the definition of tree automata and the related terminologies. In section 3 we present a polynomial time algorithm to learn dfrtas which is a slight variant of the Sakakibara's polynomial time algorithm to learn deterministic skeletal automata. In section 4 we review the definition of weak monadic second-order theory of multiple successors (WSMS for short) and introduce some necessary terminologies. In section 5 we present our main theorem and illustrate by two examples how the learner  $L_T^*$  learns a relation definable in the language associated to the WSMS from queries and counterexamples. We present two example runs of  $L_T^*$  to learn the unary predicate  $PC$  which is the set of prefix-closed sets, and the binary relation " $\subseteq$ " on finite sets.

## 2 Preliminaries

In this section, we review the definition of tree automata and the related terminologies. For details, see [9, 12, 13].

An empty string is denoted by  $\lambda$  and the power set of a set  $A$  is denoted by  $2^A$ . For a set  $A$ ,  $A^*$  denotes the free monoid generated by  $A$ . Let  $N$  be the set of non-negative integers. For  $x, y \in N$ ,  $x \succeq y$  iff there exists  $z \in N^*$  such that  $x = yz$ , and  $x \succ y$  iff  $x \succeq y$  and  $x \neq y$ .

A *ranked alphabet* is defined to be an pair  $V = \langle \Sigma, \sigma \rangle$ , where  $\Sigma$  is a finite set of symbols and  $\sigma$  is a mapping from  $\Sigma$  into  $N$ . For  $a \in \Sigma$ ,  $\sigma(a)$  is called the *rank* of  $a$ . We will denote the set of symbols of rank  $n$  by  $\Sigma_n$ , i.e.  $\Sigma_n = \sigma^{-1}(n)$ . A symbol in the set  $\Sigma_0$  is called a *constant symbol*. If we consider the symbols in  $\Sigma_n$  as function symbols, the rank of each function symbol is usually called its *arity*.

**Definition 2.1** A  $\Sigma$ -tree, or a tree over  $\Sigma$  is a mapping  $t$  from  $Dom(t)$  into  $\Sigma$ , where  $Dom(t)$  is a finite subset of  $N^*$  satisfying :

1. If  $x \in Dom(t)$  and  $x \succ y$  then  $y \in Dom(t)$ .

2. If  $yi \in \text{Dom}(t)$  for  $i \in N$  then  $yj \in \text{Dom}(t)$  for  $j \in N$ ,  $1 \leq j \leq i$ .
3. If  $t(x) = a \in \Sigma_n$  then  $xi \in \text{Dom}(t)$  for  $i \in N$ ,  $1 \leq i \leq n$ .

An element of  $\text{Dom}(t)$  is called a *node* of  $t$ . If  $t(x) = a$ , then  $a$  is said to be the *label* of the node  $x$  of  $t$ . The set of all trees over  $\Sigma$  is denoted by  $T_\Sigma$ .

**Definition 2.2** A ranked alphabet  $V = \langle \Sigma, \sigma \rangle$  uniquely determines a set  $\text{Term}(\Sigma)$  of *terms over  $\Sigma$*  defined to be the least subset of  $\Sigma^*$  satisfying :

1.  $\Sigma_0 \subseteq \text{Term}(\Sigma)$ .
2. If  $f \in \Sigma_n$  and  $t_1, \dots, t_n \in \text{Term}(\Sigma)$  then  $ft_1 \dots t_n \in \text{Term}(\Sigma)$ .

Since the finite trees over  $\Sigma$  can be identified with terms over  $\Sigma$ , we will represent trees as terms.

Let  $t \in T_\Sigma$ . A node  $p$  in  $t$  is a *terminal node* iff for all  $q \in \text{Dom}(t)$ ,  $p \not\prec q$ . While a node  $p$  in  $t$  is an *interior node* iff  $p$  is not a terminal node. The *frontier* of  $\text{Dom}(t)$  is the set of all terminal nodes in  $\text{Dom}(t)$ . The depth of  $p \in \text{Dom}(t)$  is the length of  $p$  and denoted  $\text{depth}(p)$ . For a tree  $t$ , we define the depth of  $t$  by  $\text{depth}(t) = \max\{\text{depth}(p) \mid p \in \text{Dom}(t)\}$ . For  $p \in \text{Dom}(t)$ , we define the *subtree  $t/p$  of  $t$  at  $p$*  by  $t/p(q) = t(pq)$ .

Let  $\$$  be a new symbol of rank 0 which is not included in  $\Sigma$ . Then  $T_\Sigma^\$$  denotes the set of all trees over  $\Sigma \cup \{\$\}$  which contains exactly one  $\$$ -symbol. For trees  $u \in T_\Sigma^\$$  and  $v \in T_\Sigma \cup T_\Sigma^\$$ , an operation  $\#$  to replace the terminal node labelled  $\$$  of  $u$  with  $v$  is defined as follows :

$$u\#v(p) = \begin{cases} u(p) & \text{if } p \in \text{Dom}(u) \text{ and } u(p) \neq \$, \\ v(q) & \text{if } p = rq, u(r) = \$ \text{ and } q \in \text{Dom}(v). \end{cases}$$

For subsets  $U \subseteq T_\Sigma^\$$  and  $V \subseteq T_\Sigma \cup T_\Sigma^\$$ , we define  $U\#V = \{u\#v \mid u \in U \text{ and } v \in V\}$ .

**Definition 2.3** A *deterministic frontier-to-root tree automata* (dfirta for short) is a quadruple  $M = (Q, \Sigma, \delta, F)$  consists of 1-4 as follows :

1.  $Q$  is a finite set of *states*.
2.  $\Sigma$  is a ranked alphabet with the maximal rank  $n$ .

3.  $\delta = (\delta_0, \delta_1, \dots, \delta_n)$  is the *state transition function*, where  
 $\delta_k : \Sigma_k \times (Q \cup \Sigma_0)^k \rightarrow Q$  ( $k = 1, 2, \dots, n$ ), and  
 $\delta_0(a) = a$  for  $a \in \Sigma_0$ .
4.  $F \subseteq Q$  is the set of *final states*.

The terminal symbols on the frontier are taken as *initial states*. We extend  $\delta$  to  $\mathcal{T}_\Sigma$  as usual by :

$$\delta(ft_1 \dots t_k) = \begin{cases} \delta_k(f, \delta(t_1), \dots, \delta(t_k)) & \text{if } k > 0, \\ \delta_0(f) & \text{if } k = 0. \end{cases}$$

A tree  $t$  is *accepted* by  $M$  iff  $\delta(t) \in F$ . We define the set of trees accepted by  $M$  as  $L(M) = \{t \in \mathcal{T}_\Sigma \mid \delta(t) \in F\}$ . A set  $L$  of trees is said to be *recognizable* if there exists a dfrta  $M$  such that  $L = L(M)$ .

### 3 Learning Tree Automata

In this section, we will present a slight variant of the Sakakibara's polynomial time algorithm to learn deterministic skeletal automata [9] as the one to learn deterministic frontier-to-root tree automata. For details, see [9].

#### 3.1 Closed Consistent Observation Tables

Let  $\Sigma$  be a ranked alphabet,  $A$  be a finite subset of  $\mathcal{T}_\Sigma$ , and  $B$  be a finite subset of  $\mathcal{T}_\Sigma^\$$ . A set  $A$  is *subtree-closed* if  $a \in A$  then all subtrees with depth at least 1 of  $a$  are included in  $A$ . While a set  $B$  is *prefix-closed with respect to  $A$*  if  $b \in B - \{\$ \}$  then there exists a tree  $b' \in B$  such that  $b = b' \# f a_1 \dots a_{i-1} \$ a_i \dots a_{k-1}$  for some  $f \in \Sigma_k$ ,  $a_1, \dots, a_{k-1} \in A \cup \Sigma_0$  and  $i \in N$ .

**Definition 3.1** An *observation table* is a 3-tuple  $(S, E, T)$  consists of 1-4 as follows :

1.  $S$  is a nonempty subtree-closed set of  $\Sigma$ -trees with depth at least 1.
2.  $X(S) = \{ f u_1 \dots u_k \mid f \in \Sigma_k, u_1, \dots, u_k \in S \cup \Sigma_0 \text{ and } f u_1 \dots u_k \notin S \text{ for } k \geq 1 \}$

3.  $E$  is a nonempty finite subset of  $T_\Sigma^\$$  which is prefix-closed with respect to  $S$ .
4.  $T$  is a finite function mapping  $E\#(S \cup X(S))$  to  $\{0, 1\}$ .

The interpretation of  $T$  is as follows :  $T(s) = 1$  iff  $s \in L(M)$  of the unknown dfrta  $M$ . An observation table can be visualised as a two-dimensional matrix with rows labelled by elements of  $S \cup X(S)$ , columns labelled by elements of  $E$ , and the entry for row  $s$  and column  $e$  equal to  $T(e\#s)$ . The learning algorithm uses the observation table to build a dfrta. If  $s$  is an element of  $S \cup X(S)$ ,  $row(s)$  denotes the finite function  $g$  from  $E$  to  $\{0, 1\}$  defined by  $g(e) = T(e\#s)$ .

An observation table  $(S, E, T)$  is *closed* if every  $row(x)$  of  $x \in X(S)$  is identical to some  $row(s)$  of  $s \in S$ . An observation table is called *consistent* if whenever  $s_1$  and  $s_2$  are elements of  $S$  such that  $row(s_1) = row(s_2)$ ,  $row(fu_1...u_{i-1}s_1u_i...u_{k-1}) = row(fu_1...u_{i-1}s_2u_i...u_{k-1})$  for all  $f \in \Sigma_k$ ,  $u_1, \dots, u_{k-1} \in S \cup \Sigma_0$  and  $1 \leq i \leq k$ .

Let  $(S, E, T)$  be a closed consistent observation table (CCOT for short). The *corresponding dfrta*  $M(S, E, T)$  over  $\Sigma$  constructed from  $(S, E, T)$  is defined with the state set  $Q$ , the set of final states  $F$ , and the state transition function  $\delta$  as follows :

$$\begin{aligned} Q &= \{ row(s) \mid s \in S \}, \\ F &= \{ row(s) \mid s \in S \text{ and } T(s) = 1 \}, \\ \delta_k(f, row(s_1), \dots, row(s_k)) &= row(fs_1...s_k) \\ &\quad \text{for } f \in \Sigma_k, \text{ and } s_1, \dots, s_k \in S \cup \Sigma_0, \\ \delta_0(a) &= row(a) \text{ for } a \in \Sigma_0. \end{aligned}$$

The concept of the CCOT was introduced by Angluin [1]. The following theorem is similarly proved as in [9]

**Theorem 3.1** *Let  $(S, E, T)$  be a CCOT. The dfrta  $M = M(S, E, T)$  defined above satisfies the following four conditions :*

1.  $M$  is well-defined.
2. For every  $s \in S \cup X(S)$ ,  $\delta(s) = row(s)$ .
3.  $M$  is consistent with the finite function  $T$ . That is, for every  $s \in S \cup X(S)$  and  $e \in E$ ,  $\delta(e\#s) \in F$  iff  $T(e\#s) = 1$ .

4. Suppose that  $M$  has  $n$  states. If  $M'$  is any dfrta consistent with  $T$  that has  $n$  or fewer states, then  $M'$  is isomorphic to  $M$ .  $\square$

### 3.2 The Learning Algorithm $L_T^*$

Let  $L_U$  be an unknown recognizable tree language and  $M_U$  be the minimum dfrta accepting  $L_U$ . It is assumed that the learner knows the ranked alphabet  $\Sigma$  of  $M_U$ . A *membership query* proposes a  $\Sigma$ -tree  $t$  and asks whether  $t \in L_U$ . The answer from the teacher is either *yes* or *no*. While an *equivalence query* proposes a dfrta  $M$  and asks whether  $L(M) = L_U$ . The answer is either *yes* or *no*. When the answer is *no*, a *counterexample* is also provided from the teacher. It is a  $\Sigma$ -tree  $t$  in the symmetric difference of  $L_U$  and  $L(M)$ . This learning protocol is based on Angluin's "minimally adequate teacher" in [1]. Our algorithm to learn dfrta is shown in Figure 1.

In the algorithm  $L_T^*$ , the operation "extend  $T$  to  $E\#(S \cup X(S))$  using membership queries" is the operation to extend  $T$  by asking membership queries for missing elements. It is clear that if  $L_T^*$  ever terminates, its output is a dfrta  $M$  such that  $L(M) = L_U$ . The following theorem is similarly proved as in [9]

**Theorem 3.2** *Let  $L_U$  be an unknown recognizable tree language and  $M_U$  be the minimum dfrta accepting  $L_U$ . Using membership and equivalence queries for  $L_U$ , the learning algorithm  $L_T^*$  eventually terminates and outputs a dfrta  $M$  isomorphic to  $M_U$  accepting  $L_U$ . Furthermore, if  $n$  is the number of the states of  $M_U$  and  $m$  is the maximum size of any counterexample provided by the teacher, then the total running time of  $L_T^*$  is bounded by a polynomial in  $m$  and  $n$ .  $\square$*

## 4 Weak Monadic Second-Order Theory of Multiple Successors

In this section, we review the definition of weak monadic second-order theory of multiple successors (WSMS for short) and introduce some necessary terminologies. For details, see [13]. We mainly describe how a dfrta recognizes a definable relation in the language associated to WSMS.



```

 $S := \Sigma_0$ ;  $E := \{\$ \}$ ;
Construct the initial observation table  $(S, E, T)$  using membership queries
  for all  $\Sigma$ -trees of depth at most 2;
Repeat
  While  $(S, E, T)$  is not closed or not consistent do
    If  $(S, E, T)$  is not closed then do
      Find  $s_1 \in X(S)$  s.t.  $row(s_1)$  is different from  $row(s)$  for all  $s \in S$ ;
      Add  $s_1$  to  $S$ ;
      Extend  $T$  to  $E\#(S \cup X(S))$  using membership queries
    end
    If  $(S, E, T)$  is not consistent then do
      Find  $s_1, s_2 \in S, e \in E, k \in N, f \in \Sigma_k, u_1, \dots, u_{k-1} \in S \cup \Sigma_0$ ,
        and  $i \in N$  such that  $row(s_1) = row(s_2)$  and
         $T(e\#f u_1 \dots u_{k-1} s_1 u_i \dots u_{k-1}) \neq T(e\#f u_1 \dots u_{k-1} s_2 u_i \dots u_{k-1})$ ;
      Add  $e\#f u_1 \dots u_{k-1} \$ u_i \dots u_{k-1}$  to  $E$ ;
      Extend  $T$  to  $E\#(S \cup X(S))$  using membership queries
    end
  end
  Once  $(S, E, T)$  is closed and consistent, let  $M := M(S, E, T)$ ;
  Make the conjecture  $M$  using an equivalence query proposing  $M$ ;
  If the teacher replies no with a counterexample  $t$  then do
    Add  $t$  and all its subtrees with depth at least 1 to  $S$ ;
    Extend  $T$  to  $E\#(S \cup X(S))$  using membership queries
  end
Until the teacher replies yes to the conjecture  $M$ ;
Halt and output  $M$ .

```

Figure 1: The learning algorithm  $L_T^*$ .

#### 4.1 A Monadic Second-Order Language $\mathcal{L}_k$

Let  $A_k = \{1, 2, \dots, k\}$  be an alphabet. The WSMSs are based on (1) the set  $A_k^*$ , (2)  $k$  right-successor functions,  $r_1, \dots, r_k$ , where  $r_i(w) = wi$  for all  $w \in A_k^*$  and  $\lambda i = i$  ( $1 \leq i \leq k$ ), and (3) relations and functions which can be defined recursively from the successor functions of (2). In this paper, we deal with the *weak monadic second-order theory of  $A_k^*$  with the  $k$  successor functions*. The associated language  $\mathcal{L}_k$  is a monadic second-order language consisting of the following :

- *individual variables*,  $x, y, z, x_1, y_1, z_1, \dots$ , ranging over  $A_k^*$ .
- *set variables*,  $\alpha, \beta, \alpha_1, \beta_1, \dots$ , ranging over finite subsets of  $A_k^*$ .
- *constants*  $=, \in$  with their usual interpretation.
- *binary predicate symbols*  $R_i$ ,  $i \in A_k$  interpreted as follows :  $R_i(u, v) \leftrightarrow r_i(u) = v \leftrightarrow ui = v$ .
- *propositional connectives*  $\wedge, \neg$  ; *individual quantifier*,  $\exists$  ; *set quantifier*,  $\exists$  ; *punctuation and parentheses* .

Notice that  $\vee, \forall$  and  $\rightarrow$  can be defined using  $\wedge, \neg$  and  $\exists$ . So, in the sequel, we also use  $\vee, \forall$  and  $\rightarrow$  to define relations in  $\mathcal{L}_k$ . *Atomic formulas* are those expressions of the form  $x = y$ ,  $x \in \alpha$  or  $R_i(x, y)$ .

**Definition 4.1**  $\mathcal{L}_k$ -formulas are recursively defined as follows :

1. Atomic formulas are  $\mathcal{L}_k$ -formulas.
2. If  $F$  and  $G$  are  $\mathcal{L}_k$ -formulas, then  $F \wedge G$ ,  $\neg F$ ,  $\exists x F$  and  $\exists \alpha F$  are all  $\mathcal{L}_k$ -formulas.
3. Nothing other than those that can be defined by means of a finite number of applications of the rules 1 and 2 are  $\mathcal{L}_k$ -formulas.

The *sentences* of  $\mathcal{L}_k$  are those  $\mathcal{L}_k$ -formulas including no free variables. The WSMS consists of all true sentences in  $\mathcal{L}_k$ .

**Example 4.1** Let  $PC(\alpha)$  be a unary predicate which is the set of prefix-closed subsets of  $A_2^*$ . That is,  $PC(\alpha)$  is *true* iff a set  $\alpha$  is prefix-closed.  $PC(\alpha)$  is definable in  $\mathcal{L}_2$  :

$$PC(\alpha) \stackrel{\text{def}}{\iff} \forall x ([\exists y (R_1(x, y) \wedge y \in \alpha) \vee \exists y (R_2(x, y) \wedge y \in \alpha)] \rightarrow (x \in \alpha)).$$

The weak monadic second-order theory of one successor ( the case  $k = 1$  ) is known to be decidable [3, 4, 7], that is, there is an effective procedure for deciding truth of sentences of  $\mathcal{L}_1$ . While, the weak monadic second-order theory of multiple successors is also shown to be decidable using concepts of generalized finite automata, which is equivalent to dfrta [4, 5, 13]. That is, the following theorem is known:

**Theorem 4.1** [5] *The weak monadic second-order theory of multiple successors is decidable.*  $\square$

## 4.2 Recognizing Definable Relations of $\mathcal{L}_2$

In the sequel, we will concentrate the case  $k = 2$  because the generalization for more than two successor functions offers no difficulty. The outline of the procedure to recognize a definable relation of the language  $\mathcal{L}_2$  is as follows :

1. Find a language  $\mathcal{L}'_2$  equivalent to  $\mathcal{L}_2$  which involves no individual variables or individual quantifiers.
2. Encode  $n$ -tuples of finite subsets of  $A_2^*$  as terms on the ranked alphabet  $\Sigma^n = \Sigma_0^n \cup \Sigma_2^n$ , where  $\Sigma_0^n = \{\varepsilon\}$  and  $\Sigma_2^n = \{0, 1\}^n$ .
3. Find a dfrta recognizing every definable relation of  $\mathcal{L}'_2$  interpreted under the encoding of 2 ( Existence of such a dfrta is guaranteed by theorem 4.2 stated in below ).

Now we will describe the equivalent language  $\mathcal{L}'_2$  and the encoding of  $n$ -tuples of finite subsets of  $A_2^*$ .

*The equivalent Language  $\mathcal{L}'_2$*

The individual variables of  $\mathcal{L}_2$  can be eliminated by simply replacing them with set variables which are restricted to be singleton sets. We will introduce

set variables in one-one correspondence with the individual variables. The language  $\mathcal{L}'_2$  is a monadic second-order language consisting of the following :

- *set variables*,  $\alpha_x, \alpha_y, \alpha_z, \alpha_{x_1}, \alpha_{y_1}, \alpha_{z_1}, \dots$ , ranging over singleton subsets of  $A_2^*$ .
- *set variables*,  $\alpha, \beta, \alpha_1, \beta_1, \dots$ , ranging over finite subsets of  $A_2^*$ .
- *constant*  $\subseteq$  with its usual interpretation.
- *binary predicate symbols*  $\bar{R}_i$ ,  $i \in A_i$  interpreted as follows :  $\bar{R}_i(\alpha, \beta) \leftrightarrow \alpha$  and  $\beta$  are singletons and  $\hat{r}_i(\alpha) = \beta$  (  $\hat{r}_i$  is the set function induced by  $r_i$  ).
- *propositional connectives*  $\wedge, \neg$  ; *set quantifier*,  $\exists$  ; *punctuation and parentheses* .

The formation rules for  $\mathcal{L}'_2$  are like those of  $\mathcal{L}_2$  except that individual variables are not involved.

Next we define a translation  $\tau$  from  $\mathcal{L}_2$  to  $\mathcal{L}'_2$ . We introduce the predicate  $SI(\alpha)$  which is the set of singleton subsets.  $SI$  is definable in  $\mathcal{L}'_2$  :

$$SI(\alpha) \stackrel{\text{def}}{\leftrightarrow} \forall \beta [\beta \supseteq \alpha \rightarrow (\alpha \supseteq \beta \vee \forall \beta_1 (\beta \supseteq \beta_1))] \wedge \exists \beta_1 (\neg \alpha \supseteq \beta_1).$$

Then the translation is defined as shown in Figure 2. It should be clear that if  $s$  is any sentence in  $\mathcal{L}_2$  then  $s$  is true iff  $\tau(s)$  is true.

*The encoding of finite subsets of  $A_2^*$*

Here, we will be working with the ranked alphabet  $\Sigma^1 = \Sigma_0^1 \cup \Sigma_2^1$ , where  $\Sigma_0^1 = \{\varepsilon\}$  and  $\Sigma_2^1 = \{0, 1\}$ . We define terms as functions from finite prefix closed subsets of  $A_2^*$  into  $\Sigma^1$ . For example, the term  $t = 001\varepsilon\varepsilon 1\varepsilon\varepsilon$  shown in Figure 3 (b) corresponds to the function from a set  $S = \{\lambda, 1, 2, 11, 12, 21, 22, 111, 112\}$  into  $\Sigma^1$  defined by the corresponding position of the trees in Figure 3, e.g.  $t(\lambda) = 0$ ,  $t(11) = 1$ ,  $t(112) = \varepsilon$ .

If we look at the inverse image of some function symbol, we are able to associate a finite subset of  $A_2^*$  with a term. For example, as can be seen from Figure 3,  $t^{-1}(1) = \{2, 11\}$  and  $t^{-1}(0) = \{\lambda, 1\}$ .

	$s$	$\tau(s)$
atomic formulas	$x = y$	$(\alpha_x \supseteq \alpha_y) \wedge SI(\alpha_x) \wedge SI(\alpha_y)$
	$x \in \alpha$	$(\alpha_x \supseteq \alpha) \wedge SI(\alpha_x)$
	$R_i(x, y)$	$R_i(\alpha_x, \alpha_y)$
extension	$F \wedge G$	$\tau(F) \wedge \tau(G)$
	$\neg F$	$\neg \tau(F)$
	$\exists x F$	$\exists \alpha_x [SI(\alpha_x) \wedge \tau(F)]$
	$\exists \alpha F$	$\exists \alpha \tau(F)$

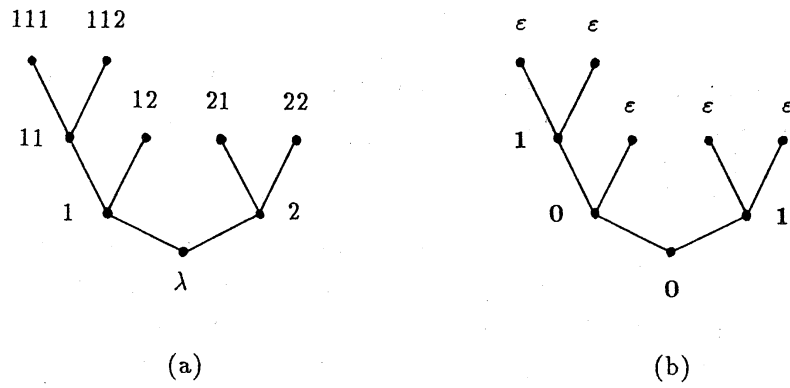
Figure 2: The translation  $\tau$  from  $\mathcal{L}_2$  to  $\mathcal{L}'_2$ .

Figure 3: A term as a function

Thus, a term in  $T_{\Sigma^1}$  can be viewed as a characteristic function of a finite subset of  $A_2^*$ , that is, the set associated with such a term  $t$  is  $t^{-1}(1)$ . An inductive definition of the mapping  $c$  which assigns to each  $t \in T_{\Sigma^1}$  a finite subset  $c(t) \subseteq A_2^*$  is as follows :

1.  $c(\varepsilon) = \emptyset$ ,
2.  $c(0t_1t_2) = \hat{l}_1c(t_1) \cup \hat{l}_2c(t_2)$   
 $c(1t_1t_2) = \hat{l}_1c(t_1) \cup \hat{l}_2c(t_2) \cup \{\lambda\}$ ,

where  $l_i$  is the left successor by the symbol  $i$ , i.e.  $l_i(w) = iw$ , and  $\hat{l}_i$  is the set function induced by  $l_i$ .

It is inductively proved that the function  $c$  is onto  $2^{A_2^*}$ . Though,  $c$  is not one-one. In general,  $c(t) = c(t')$  iff  $t$  and  $t'$  differ only by subterms in the function symbol  $0$ . Thus, we define the encoding  $e(\alpha)$  of  $\alpha$  to be the term in  $c^{-1}(\alpha)$  in which  $0\varepsilon\varepsilon$  is not a subterm. The following proposition is known :

**Proposition 4.1** [13] *The set of encodings of finite subsets of  $A_2^*$  is recognizable.*  $\square$

*The encoding of  $n$ -tuples of finite subsets of  $A_2^*$*

We now define the encoding of  $n$ -tuples of finite sets. We will be working with the ranked alphabet  $\Sigma^n = \Sigma_0^n \cup \Sigma_2^n$ , where  $\Sigma_0^n = \{\varepsilon\}$  and  $\Sigma_2^n = \{0, 1\}^n$ . Define the mapping  $p_i : \{0, 1\}^n \rightarrow \{0, 1\}$  ( $i = 1, \dots, n$ ) by  $p_i(a_1, \dots, a_n) = a_i$ . Then, we extend  $p_i$  to a projection  $\bar{p}_i : T_{\Sigma^n} \rightarrow T_{\Sigma^1}$ . We first define a mapping  $c_n$  from  $T_{\Sigma^n}$  to  $(2^{A_2^*})^n$  by

$$c_n(t) = (c\bar{p}_1t, \dots, c\bar{p}_nt).$$

Again, the encoding of an  $n$ -tuple  $\bar{\alpha} = (\alpha_1, \dots, \alpha_n)$  is chosen to be the term  $e(\bar{\alpha})$  in  $c_n^{-1}(\bar{\alpha})$  in which  $0\dots 0\varepsilon\varepsilon$  is not a subterm. For example, a term in Figure 4 is equal to  $e(\{\lambda, 1, 11\}, \{1, 21\})$ . The following proposition is known :

**Proposition 4.2** [13] *The set of encodings of  $n$ -tuples of finite subsets of  $A_2^*$  is recognizable.*  $\square$

A relation  $R$  on finite subsets of  $A_2^*$ , i.e.  $R \subseteq (2^{A_2^*})^n$ , is said to be *recognizable*

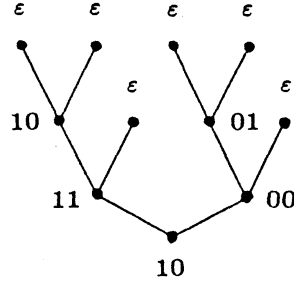


Figure 4: An encoding of a pair of finite sets.

iff  $\hat{e}R$  is a recognizable tree language, where  $\hat{e}$  is the set function induced by  $e$ . The following theorem is known :

**Theorem 4.2** [13] *If  $R$  is a relation definable in  $\mathcal{L}'_2$  then  $R$  is recognizable.*

□

## 5 Learning Weak Monadic Second-Order Logical Formulas

In this section, we present our main theorem concerning the learning of a relation definable in  $\mathcal{L}_k$ . Then we illustrate by two examples how the learner  $L_T^*$  learns a relation definable in  $\mathcal{L}_2$  from queries and counterexamples.

### 5.1 Main Theorem

Immediately from theorems 3.2 and 4.2, we obtain the following main theorem :

**Theorem 5.1** *Let  $R$  be a relation definable in  $\mathcal{L}_k$  and  $M_R$  be the minimum dfrta accepting  $\hat{e}R$ . Using membership and equivalence queries for  $\hat{e}R$ , the learning algorithm  $L_T^*$  eventually terminates and outputs a dfrta  $M$  isomorphic to  $M_R$  accepting  $\hat{e}R$ . Furthermore, if  $n$  is the number of the states of  $M_R$  and  $m$  is the maximum size of any counterexample provided by the teacher, then the total running time of  $L_T^*$  is bounded by a polynomial in  $m$  and  $n$ . □*

**Corollary 5.1** *Let  $R$  be a relation definable in  $\mathcal{L}_k$ . There is an algorithm to learn  $R$  using membership and equivalence queries that runs in time polynomial in the number of states of the minimum dfrrta accepting  $\hat{e}R$  and the maximum length of any counterexample provided by the teacher.*  $\square$

We assume that the learner  $L_T^*$  knows the following things :

1. the ranked alphabet of the dfrrta to be learned,
2. the valid encoding of finite subsets of  $A_k^*$ .

Thus, for a term  $t$  which is not a valid encoding for any tuple of finite subsets of  $A_k$  ( i.e.  $t$  has  $0\dots 0\varepsilon\varepsilon$  as a subterm ),  $L_T^*$  automatically fills in the corresponding entry of the observation table with "0" without asking to the teacher.

## 5.2 Two Examples

In this subsection, we will illustrate by two examples how the learner  $L_T^*$  learns a relation definable in  $\mathcal{L}_2$  from queries and counterexamples. In examples 5.1 and 5.2, we present example runs of  $L_T^*$  to learn the unary predicate  $PC$  of Example 4.1 and the binary relation " $\subseteq$ ", respectively.

**Example 5.1** ( Learning of the unary predicate  $PC$  of Example 4.1 )

As mentioned above, we assume that the learner  $L_T^*$  knows the ranked alphabet  $\Sigma^1 = (\Sigma_0^1, \Sigma_2^1)$ , where  $\Sigma_0^1 = \{\varepsilon\}$  and  $\Sigma_2^1 = \{0, 1\}$ . It is easy to see that

$$L_{PC} = \{t \in T_{\Sigma^1} \mid t = e(\alpha) \text{ for some } \alpha \subseteq A_2^*, \text{ and if } v \prec u \text{ and } t(u) = 1 \text{ then } t(v) = 1.\}$$

is the unknown recognizable tree language to be learned.

The learner  $L_T^*$  constructs the initial observation table  $OT_1$  shown in Figure 5 (a) using membership queries for all  $\Sigma^1$ -trees with depth at most 2. Since  $0\varepsilon\varepsilon$  is an invalid encoding,  $L_T^*$  automatically fills in the entry in the row 2 of  $OT_1$  with '0' as mentioned above. In order to fill in the entries in rows 1 and 3,  $L_T^*$  proposes to the teacher two membership queries asking " $\varepsilon \in$



	$OT_1$	$\$$
1	$\varepsilon$	1
2	$0\varepsilon\varepsilon$	0
3	$1\varepsilon\varepsilon$	1

(a)

	$OT_2$	$\$$
1	$\varepsilon$	1
2	$0\varepsilon\varepsilon$	0
3	$1\varepsilon\varepsilon$	1
4	$0\varepsilon 0\varepsilon\varepsilon$	0
5	$00\varepsilon\varepsilon\varepsilon$	0
6	$00\varepsilon\varepsilon 0\varepsilon\varepsilon$	0
7	$1\varepsilon 0\varepsilon\varepsilon$	0
8	$10\varepsilon\varepsilon\varepsilon$	0
9	$10\varepsilon\varepsilon 0\varepsilon\varepsilon$	0

(b)

$\delta_0(\varepsilon) = q_0,$   

$\delta_2$	0	1
$q_0, q_0$	$q_1$	$q_0$
$q_0, q_1$	$q_1$	$q_1$
$q_1, q_0$	$q_1$	$q_1$
$q_1, q_1$	$q_1$	$q_1$

(c)

Figure 5: The learning of the predicate  $PC$ , (a)  $OT_1$ ,  $S = \{\varepsilon\}$ ,  $E = \{\$\}$ , (b)  $OT_2$ ,  $S = \{\varepsilon, 0\varepsilon\varepsilon\}$ ,  $E = \{\$\}$ , (c) the conjecture  $M$  of  $L_T^*$ .

$L_{PC} ?$ " and " $1\varepsilon\varepsilon \in L_{PC} ?$ ", respectively. Each of these queries respectively corresponds to asking " $\emptyset$  is prefix-closed?" and " $\{\lambda\}$  is prefix-closed?".

Since  $OT_1$  is not closed,  $L_T^*$  adds  $0\varepsilon\varepsilon$  to  $S$  and construct the second observation table  $OT_2$  shown in Figure 5 (b) by extending  $OT_1$ . Notice that when  $L_T^*$  constructs  $OT_2$  from  $OT_1$ ,  $L_T^*$  does not propose any membership query to the teacher. This is because every term in the rows from 4 to 9 of  $OT_2$  has  $0\varepsilon\varepsilon$  as a subterm and thus they are invalid encodings. Hence,  $L_T^*$  automatically fills in the entries in the rows from 4 to 9 with '0'. Since  $OT_2$  is a CCOT,  $L_T^*$  makes the first conjecture  $M$  presented in Figure 5 (c) and proposes an equivalence query. The initial state of  $M$  is  $q_0$  and the final state is also  $q_0$ . Since  $M$  is a correct dfrta for  $L_{PC}$ , the teacher replies to this conjecture with *yes*. Thus,  $L_T^*$  halts and outputs  $M$ .

The total number of membership queries during this example run is 2 ( $L_T^*$  asked to the teacher only about rows 1 and 3 of  $OT_1$ ). And  $L_T^*$  makes 1 correct conjecture only.

### Example 5.2 ( Learning of the binary relation " $\subseteq$ " )

We assume that the learner  $L_T^*$  knows the ranked alphabet  $\Sigma^2 = (\Sigma_0^2, \Sigma_2^2)$ , where  $\Sigma_0^2 = \{\varepsilon\}$  and  $\Sigma_2^2 = \{00, 01, 10, 11\}$ . Notice that in this example,  $\varepsilon$  is not a valid encoding for any pair of finite subsets of  $A_2^*$ . It can be seen that

$$L_{SUB} = \{t \in T_{\Sigma^2} \mid t = e(\alpha) \text{ for some } \alpha \subseteq A_2^*, \text{ and there is no node in } t$$

labelled by 10.}

is the unknown recognizable tree language to be learned.

The learner constructs the initial observation table  $OT_1$  shown in Figure 6 (a) using membership queries for all  $\Sigma^2$ -trees with depth at most 2. Since  $OT_1$  is not closed,  $L_T^*$  adds  $01\varepsilon\varepsilon$  to  $S$  and constructs the second observation table  $OT_2$  shown in Figure 6 (b) by extending  $OT_1$ . Since  $OT_2$  is a CCOT,  $L_T^*$  makes the first conjecture  $M_1$  presented in Figure 7. The initial state of  $M_1$  is  $q_0$  and the final state is  $q_1$ .

Since  $M_1$  is not a correct dfrta for  $L_{SUB}$ , the teacher replies to this conjecture with *no* and provides a counterexample. Let us assume that the teacher provides the counterexample  $01\varepsilon10\varepsilon\varepsilon$ . It is not in  $L_{SUB}$  but accepted by  $M_1$ . Then,  $L_T$  adds the counterexample and all its subtrees to  $S$  and constructs the third observation table  $OT_3$  in Figure 8 by extending  $OT_2$ .

This time  $OT_3$  is closed but is not consistent. So,  $L_T^*$  adds  $11\varepsilon\varepsilon$  to  $E$  and constructs the fourth observation table  $OT_4$  shown in Figure 9 by extending  $OT_3$ . Since  $OT_4$  is a CCOT,  $L_T^*$  makes the second conjecture  $M_2$  shown in Figure 10. The initial state of  $M_2$  is  $q_0$  and the final state is  $q_1$ . Since  $M_2$  is a correct dfrta for  $L_{SUB}$ , the teacher replies to this conjecture with *yes*. Thus,  $L_T^*$  halts and outputs  $M_2$ .

The total number of membership queries during this example run is 130. And  $L_T^*$  makes 1 incorrect conjecture and 1 correct conjecture.

## 6 Concluding Remarks

In this paper, we have shown that weak monadic second-order logical formulas can be learned in polynomial time using membership queries and equivalence queries. Since the learner  $L_T^*$  knows the valid encodings of finite sets, the number of membership queries used by  $L_T^*$  becomes very small in some cases such as in Example 4.1. Though, it is still an open problem to substantially reduce the number of membership queries used by  $L_T^*$ . On the other hand, the numbers of equivalence queries used by  $L_T^*$  in Examples 5.1 and 5.2 are both quite small compared with the cases in the Shapiro's framework [11].

There is an interesting and important extension of our framework to be performed. Rabin showed that the monadic second-order theory of multiple

$OT_1$	\$
$\varepsilon$	0
00 $\varepsilon\varepsilon$	0
01 $\varepsilon\varepsilon$	1
10 $\varepsilon\varepsilon$	0
11 $\varepsilon\varepsilon$	1

(a)

$OT_2$	\$
$\varepsilon$	0
01 $\varepsilon\varepsilon$	1
00 $\varepsilon\varepsilon$	0
10 $\varepsilon\varepsilon$	0
11 $\varepsilon\varepsilon$	1
00 $\varepsilon$ 01 $\varepsilon\varepsilon$	1
00 01 $\varepsilon\varepsilon\varepsilon$	1
00 01 $\varepsilon\varepsilon$ 01 $\varepsilon\varepsilon$	1
01 $\varepsilon$ 01 $\varepsilon\varepsilon$	1
01 01 $\varepsilon\vare\varepsilon$	1
01 01 $\varepsilon\varepsilon$ 01 $\varepsilon\varepsilon$	1
10 $\varepsilon$ 01 $\varepsilon\vare\varepsilon$	0
10 01 $\vare\vare\vare\vare\vare\vare\vare$	0
10 01 $\vare\vare\vare\vare\vare\vare\vare$ 01 $\vare\vare\vare\vare\vare\vare\vare$	0
11 $\vare\vare\vare\vare\vare\vare\vare$ 01 $\vare\vare\vare\vare\vare\vare\vare$	1
11 01 $\vare\vare\vare\vare\vare\vare\vare$	1
11 01 $\vare\vare\vare\vare\vare\vare\vare$ 01 $\vare\vare\vare\vare\vare\vare\vare$	1

(b)

Figure 6: The first and second observation tables, (a)  $OT_1$ ,  $S = \{\varepsilon\}$ ,  $E = \{\$\}$ , (b)  $OT_2$ ,  $S = \{\varepsilon, 01\varepsilon\vare\vare\vare\vare\vare\vare\}$ ,  $E = \{\$\}$ .

$$\delta_0(\varepsilon) = q_0,$$

$\delta_2$	00	01	10	11
$q_0, q_0$	$q_0$	$q_1$	$q_0$	$q_1$
$q_0, q_1$	$q_1$	$q_1$	$q_0$	$q_1$
$q_1, q_0$	$q_1$	$q_1$	$q_0$	$q_1$
$q_1, q_1$	$q_1$	$q_1$	$q_0$	$q_1$

Figure 7: The first conjecture  $M_1$  of  $L_T^*$ .

$OT_3$ ( part 1 )	\$	$OT_3$ ( part 2 )	\$
$\varepsilon$	0	$10\varepsilon10\varepsilon\varepsilon$	0
$01\varepsilon\varepsilon$	1	$10\ 10\varepsilon\varepsilon\varepsilon$	0
$10\varepsilon\varepsilon$	0	$10\ 01\varepsilon\varepsilon10\varepsilon\varepsilon$	0
$01\varepsilon10\varepsilon\varepsilon$	0	$10\ 10\varepsilon\varepsilon01\varepsilon\varepsilon$	0
$00\varepsilon\varepsilon$	0	$10\ 10\varepsilon\varepsilon10\varepsilon\varepsilon$	0
$10\varepsilon\varepsilon$	0	$10\ 01\varepsilon10\varepsilon\varepsilon10\varepsilon\varepsilon$	0
$11\varepsilon\varepsilon$	1	$10\ 10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$00\varepsilon01\varepsilon\varepsilon$	1	$10\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$00\ 01\varepsilon\varepsilon\varepsilon$	1	$10\ 01\varepsilon10\varepsilon\varepsilon\varepsilon$	0
$00\ 01\varepsilon\varepsilon01\varepsilon\varepsilon$	1	$10\ 01\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$01\varepsilon01\varepsilon\varepsilon$	1	$10\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon\varepsilon$	0
$01\ 01\varepsilon\varepsilon\varepsilon$	1	$10\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$01\ 01\varepsilon\varepsilon01\varepsilon\varepsilon$	1	$11\varepsilon10\varepsilon\varepsilon$	0
$10\varepsilon01\varepsilon\varepsilon$	0	$11\ 10\varepsilon\varepsilon\varepsilon$	0
$10\ 01\varepsilon\varepsilon\varepsilon$	0	$11\ 01\varepsilon\varepsilon10\varepsilon\varepsilon$	0
$10\ 01\varepsilon\varepsilon01\varepsilon\varepsilon$	0	$11\ 10\varepsilon\varepsilon01\varepsilon\varepsilon$	0
$11\varepsilon01\varepsilon\varepsilon$	1	$11\ 10\varepsilon\varepsilon10\varepsilon\varepsilon$	0
$11\ 01\varepsilon\varepsilon\varepsilon$	1	$11\ 01\varepsilon10\varepsilon\varepsilon10\varepsilon\varepsilon$	0
$11\ 01\varepsilon\varepsilon01\varepsilon\varepsilon$	1	$11\ 10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$00\varepsilon10\varepsilon\varepsilon$	0	$11\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$00\ 10\varepsilon\varepsilon\varepsilon$	0	$11\ 01\varepsilon10\varepsilon\varepsilon\varepsilon$	0
$00\ 01\varepsilon\varepsilon10\varepsilon\varepsilon$	0	$11\ 01\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$00\ 10\varepsilon\varepsilon01\varepsilon\varepsilon$	0	$11\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon\varepsilon$	0
$00\ 10\varepsilon\varepsilon10\varepsilon\varepsilon$	0	$11\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0
$00\ 01\varepsilon10\varepsilon\varepsilon10\varepsilon\varepsilon$	0		
$00\ 10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$00\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$00\ 01\varepsilon10\varepsilon\varepsilon\varepsilon$	0		
$00\ 01\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$00\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon\varepsilon$	0		
$00\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$01\varepsilon10\varepsilon\varepsilon$	0		
$01\ 10\varepsilon\varepsilon\varepsilon$	0		
$01\ 01\varepsilon\varepsilon10\varepsilon\varepsilon$	0		
$01\ 10\varepsilon\varepsilon01\varepsilon\varepsilon$	0		
$01\ 10\varepsilon\varepsilon10\varepsilon\varepsilon$	0		
$01\ 01\varepsilon10\varepsilon\varepsilon10\varepsilon\varepsilon$	0		
$01\ 10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$01\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$01\ 01\varepsilon10\varepsilon\varepsilon\varepsilon$	0		
$01\ 01\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		
$01\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon\varepsilon$	0		
$01\ 01\varepsilon10\varepsilon\varepsilon01\varepsilon10\varepsilon\varepsilon$	0		

Figure 8: The third observation table  $OT_3$ ,  $S = \{\varepsilon, 01\varepsilon\varepsilon, 10\varepsilon\varepsilon, 01\varepsilon10\varepsilon\varepsilon\}$ ,  $E = \{\$\}$ .

$OT_4$ ( part 1 )	\$	11\$ $\epsilon$
$\epsilon$	0	1
01 $\epsilon\epsilon$	1	1
10 $\epsilon\epsilon$	0	0
01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
00 $\epsilon\epsilon$	0	0
10 $\epsilon\epsilon$	0	1
11 $\epsilon\epsilon$	1	1
00 $\epsilon$ 01 $\epsilon\epsilon$	1	1
00 01 $\epsilon\epsilon\epsilon$	1	1
00 01 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	1	1
01 $\epsilon$ 01 $\epsilon\epsilon$	1	1
01 01 $\epsilon\epsilon\epsilon$	1	1
01 01 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	1	1
10 $\epsilon$ 01 $\epsilon\epsilon$	0	0
10 01 $\epsilon\epsilon\epsilon$	0	0
10 01 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
11 $\epsilon$ 01 $\epsilon\epsilon$	1	1
11 01 $\epsilon\epsilon\epsilon$	1	1
11 01 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	1	1
00 $\epsilon$ 10 $\epsilon\epsilon$	0	0
00 10 $\epsilon\epsilon\epsilon$	0	0
00 01 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
00 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
00 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
00 01 $\epsilon$ 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
00 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
00 $\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
00 01 $\epsilon$ 10 $\epsilon\epsilon\epsilon$	0	0
00 01 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
00 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
00 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
01 10 $\epsilon\epsilon\epsilon$	0	0
01 01 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
01 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
01 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
01 01 $\epsilon$ 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
01 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
01 $\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
01 01 $\epsilon$ 10 $\epsilon\epsilon\epsilon$	0	0
01 01 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
01 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
01 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0

$OT_4$ ( part 2 )	\$	11\$ $\epsilon$
10 $\epsilon$ 10 $\epsilon\epsilon$	0	0
10 10 $\epsilon\epsilon\epsilon$	0	0
10 01 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
10 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
10 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
10 01 $\epsilon$ 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
10 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
10 $\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
10 01 $\epsilon$ 10 $\epsilon\epsilon\epsilon$	0	0
10 01 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
10 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
10 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
11 $\epsilon$ 10 $\epsilon\epsilon$	0	0
11 10 $\epsilon\epsilon\epsilon$	0	0
11 01 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
11 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
11 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
11 01 $\epsilon$ 10 $\epsilon\epsilon$ 10 $\epsilon\epsilon$	0	0
11 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
11 $\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
11 01 $\epsilon$ 10 $\epsilon\epsilon\epsilon$	0	0
11 01 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0
11 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon\epsilon$	0	0
11 01 $\epsilon$ 10 $\epsilon\epsilon$ 01 $\epsilon$ 10 $\epsilon\epsilon$	0	0

Figure 9: The fourth observation table  $OT_4$ ,  $S = \{\epsilon, 01\epsilon\epsilon, 10\epsilon\epsilon, 01\epsilon 10\epsilon\epsilon\}$ ,  $E = \{\$, 11\epsilon\}$ .

$$\delta_0(\epsilon) = q_0,$$

$\delta_2$	00	01	10	11
$q_0, q_0$	$q_2$	$q_1$	$q_2$	$q_1$
$q_0, q_1$	$q_1$	$q_1$	$q_2$	$q_1$
$q_0, q_2$	$q_2$	$q_2$	$q_2$	$q_2$
$q_1, q_0$	$q_1$	$q_1$	$q_2$	$q_1$
$q_1, q_1$	$q_1$	$q_1$	$q_2$	$q_1$
$q_1, q_2$	$q_2$	$q_2$	$q_2$	$q_2$
$q_2, q_0$	$q_2$	$q_2$	$q_2$	$q_2$
$q_2, q_1$	$q_2$	$q_2$	$q_2$	$q_2$
$q_2, q_2$	$q_2$	$q_2$	$q_2$	$q_2$

Figure 10: The second conjecture  $M_2$  of  $L_T^*$ .

successors ( SMS for short ) is decidable [8]. Many interesting notions in various fields of mathematics such as topology and Boolean algebra can be defined in SMS. So, it is to be expected that a polynomial time algorithm to learn SMS formula may be developed. But in order to achieve this, a polynomial time algorithm to learn  $\omega$ -tree automata is needed because Rabin's decidability results are based on the recognition problem of tree automata on infinite trees. This is the subject for the further research.

## Acknowledgments

The author would like to express his sincere thanks to Professor Hiroshi Noguchi of Waseda University for his kind advice. He also thanks Professors Takeo Yaku and Akeo Adachi of Tokyo Denki University for their valuable suggestions.

## References

- [1] Angluin, D. : "Learning Regular Sets from Queries and Counterexamples", *Information and Computation*, Vol.75 (1987), pp.87-106.

- [2] Angluin, D. : "Learning with Hints", in Proc. of the First Workshop on Computational Learning Theory, Morgan Kaufmann (1988).
- [3] Büchi, J. R., and Elgot, C. C. : "Decision Problems of Weak Second Order Arithmetics and Finite Automata", Abstract 553-112, *Notices Amer. Math. Soc.* 5 (1958), p.834.
- [4] Büchi, J. R. : "Weak Second-Order Arithmetics and Finite Automata", University of Michigan, Logic of Computers Group Technical Report, September 1959; *Z. Math. Logik Grundlagen Math.* 6 (1960), pp.66-92.
- [5] Doner, J. E. : "Decidability of the Weak Second Order Theory of Two Successors", Abstract 65T-468, *Notices Amer. Math. Soc.* 12 (1965), p.819 ; erratum, *ibid.* 13 (1966), p.513.
- [6] Doner, J. E. : "Tree Acceptors and Some of Their Applications", *JCSS*, Vol.4 (1970), pp.406-451.
- [7] Elgot, C. C. : "Decision Problems of Finite Automata Design and Related Arithmetics", *Trans. Amer. Math. Soc.*, Vol.98 (1961), pp.21-51.
- [8] Rabin, M. O. : "Decidability of Second-Order Theories and Automata on Infinite Trees", *Trans. Amer. Math. Soc.*, Vol.141 (1969), pp.1-35.
- [9] Sakakibara, Y. : "Learning Context-Free Grammars from Structural Data in Polynomial Time", in Proc. of the First Workshop on Computational Learning Theory, Morgan Kaufmann (1988).
- [10] Sakakibara, Y. : "An Efficient Learning of Context-Free Grammars for Bottom-Up Parsers", in Proc. of FGCS'88 (1988).
- [11] Shapiro, E. : "Inductive Inference of Theories From Facts", Research Report 192, Yale University, Department of Computer Science (1981).
- [12] Thatcher, J. W. : "Tree Automata : An Informal Survey" in Aho, A. V., Ed., *Currents in the Theory of Computing*, Prentice-Hall (1973).
- [13] Thatcher, J. W. and Wright, J. B. : "Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic", *Math. Syst. Theory*, Vol.2 (1968), pp.57-81.

- [14] Valiant L. G. : "A Theory of the Learnable", *Comm. ACM*, Vol.27 (1984), pp.1134-1142.